RESTAPI

REST API

Current (RPC)	REST
GET /api/status	GET api/sessions (It might make sense for this to return only the current session, but in theory it would return all sessions that the current session is allowed to access, so for an administrator, it might return all open sessions. An individual session would be accessed at GET api/sessions/SESSIONID.)
POST /api/login token=API_TOKEN	POST api/sessions?token=API_TOKEN
GET /api/logout	DELETE api/sessions/SESSIONID or DELETE api/sessions?session=SESSIONID (get SESSIONID from api/sessions)
GET /api /get_msgs	GET api/users/USERID/messages (get USERID from api/session)
GET /api /wait_for_msgs	GET api/users/USERID/messages (long-poll?)
	GET api/messages/MESSAGEID Gets a particular message.
POST /api /send_msg message=messagebody via=client tags=tags metadata=XML_data replyto=message_id	POST api/messages?message=MESSAGE_BODY&via=CLIENT&tags=TAGS&metadata=XML&replyto=MESSAGEID
	PUT api/messages/MESSAGEID (payload the same as POST)
	DELETE api/messages/MESSAGEID
GET /api /get_following	GET api/users/USERID/followees
GET /api /get_followers	GET api/users/USERID/followers
POST /api /follow user=id_of_user	POST api/users/USERID/followees/USERID2 or POST api/users/USERID/followees?user=USERID2
POST /api /unfollow user=id_of_user	DELETE api/users/USERID/followees/USERID2 or DELETE api/users/USERID/followees?user=USERID2
GET /api /all_users	GET api/users
GET /api /get_tagcloud numTags=optional_no_of_tags	GET api/tags (This doesn't really seem like an appropriate API method. It should really return all of the tags, or user-specific tags (GET api/tags/USERID) and let the front-end decide what to do with it.)
GET /api /get_tracking	GET api/users/USERID/tracks
POST /api /add_tracking track=text	POST api/users/USERID/tracks?track=TEXT_TO_TRACK
POST /api /remove_tracking trackid=id_of_tracking_item	DELETE api/users/USERID/tracks/TRACKID
GET /api /get_conversation conversationid=Conversation_id	GET api/conversations/CONVERSATIONID

GET /api /get_actions	GET api/users/USERID/actions (Actions probably don't make sense outside of the context of a specific user.)
POST /api /add_action name=name test=trigger action=action	POST api/users/USERID/actions?name=NAME&test=TEST&action=ACTION
POST /api /enable_action id=action_id enabled=true|false	PUT api/users/USERID/actions/ACTIONID?enabled=true|false (This is actually a general outlet to update any attribute of an action, including whether or not it is enabled.)
POST /api /delete_action actionid=action_id	DELETE api/users/USERID/actions/ACTIONID

One point to note is that most HTTP clients do not currently support the "PUT" or "DELETE" methods, so these have to be simulatedthrough POST methods with an extra parameter. I think that because of the close mapping to resource verbs, is worth using these methods inthe specification and defining the simulation method for the entire APIseparately.
The above is based on a rough object hierarchy as follows:

- ESME API instance (api/)
    - Sessions (api/sessions)
    - Users (api/users)
        - Messages posted by a user (api/users/USERID/messages)
        - Users followed by a user (api/users/USERID/followees)
        - Users following a user (api/users/USERID/followers)
        - Trackers belonging to a user (api/users/USERID/tracks)
        - Actions belonging to a user (api/users/USERID/actions)
    - Messages (api/messages)
    - Tags (api/tags)
    - Conversations (api/conversations)

Each of these bullets represents a set of objects. The resource representing an individual object lives at api/objects/OBJECTID. For example, api/sessions /SESSIONID. As much as is reasonable, one would expect to be able to GET (read), POST (create), PUT (update/amend), or DELETE (delete) any individual member of each of these object sets. Going through each of these objects to ask what it would mean to create, read, update, or delete that object may reveal holes in the existing API, some of which I have filled in above.